DOI: 10.1080/18756891.2015.1129574

Adaptive Input Selection and Evolving Neural Fuzzy Networks Modeling

Alisson Marques Silva *
CEFET-MG Campus Divinopolis
Federal Center of Technological Education of Minas Gerais
Rua Alvares de Azevedo, 400, 35503-822, Divinopolis, MG, Brazil,
E-mail: alissonmarques@gmail.com
www.cefetmg.br

Walmir Caminhas, Andre Lemos
Federal University of Minas Gerais
Graduate Program in Electrical Engineering
Avenida Antonio Carlos, 6627, 31270-901, Belo Horizonte, MG, Brazil
E-mail: caminhas@cpdee.ufmg.br, andrepl@cpdee.ufmg.br
www.cpdee.ufmg.br

Fernando Gomide
University of Campinas
School of Electrical and Computer Engineering
Avenida Albert Einstein, 400, 13083-852, Campinas, SP, Brazil
E-mail: gomide@dca.fee.unicamp.br
www.dca.fee.unicamp.br

Received 11 February 2015

Accepted 26 October 2015

Abstract

This paper suggests an evolving approach to develop neural fuzzy networks for system modeling. The approach uses an incremental learning procedure to simultaneously select the model inputs, to choose the neural network structure, and to update the network weights. *Candidate* models with larger and smaller number of input variables than the *current* model are constructed and tested concurrently. The procedure employs a statistical test in each learning step to choose the best model amongst the *current* and *candidate* models. Membership functions can be added or deleted to adjust input space granulation and the neural network structure. Granulation and structure adaptation depend of the modeling error. The weights of the neural networks are updated using a gradient-descent algorithm with optimal learning rate. Prediction and nonlinear system identification examples illustrate the usefulness of the approach. Comparisons with state of the art evolving fuzzy modeling alternatives are performed to evaluate performance from the point of view of modeling error. Simulation results show that the evolving adaptive input selection modeling neural network approach achieves as high as, or higher performance than the remaining evolving modeling methods.

Keywords: Evolving Neural Fuzzy Network, Input Selection, Neo-Fuzzy Neuron, Adaptive Modeling, Prediction, Nonlinear System Identification.

* Corresponding author: Alisson Marques Silva - CEFET-MG - Campus Divinopolis, Rua Alvares de Azevedo, 400, 35503-822, Divinopolis, MG, Brazil. Tel: +55 37 3229 1150. Fax: +55 37 3229 1154. E-mail: alissonmarques@gmail.com, alisson@div.cefetmg.br.

1. Introduction

Evolving fuzzy systems constitute a class of systems whose structure and parameters can be adapted concurrently in a stepwise manner using data streams. Adaptation proceeds continuously and gradually by means of incremental learning. Incremental learning enables fast processing with low storage cost because samples in data streams are processed only once and can be discarded[1]. While learning enables continuous and gradual knowledge update changing the structure and parameters of models, it maintains the relevant knowledge of objects learned so far[2]. A limitation of the current evolving fuzzy modeling approaches concerns the non-flexibility to select the input variables as the system structure and parameters are adapted. Often, the input variables are chosen using *a priori* knowledge or a selection technique. Once chosen, the input variables remain the same[3].

A major issue in evolving systems research is how to incorporate mechanisms for input variables selection during the incremental learning process without causing damages or discontinuities in the learning process[3, 4]. Ideas to introduce adaptive selection methods have been presented in Refs. 5 and 6 for classification and in Ref. 7 for system identification.

More specifically, Ref. 5 proposes a classifier whose input variables selection scheme is part of the learning algorithm. This method creates and assigns relevance weights to a set of candidate variables. The n most relevant are selected as input variables. The relevance of the input variables are updated at each learning step, but once chosen, the number of model input variables remains fixed.

Recently, a similar incremental scheme to select input variable was proposed in Ref. 6 as part of the learning algorithm of the evolving fuzzy classifier FLEXFIS-Class[8]. The scheme also assigns relevance weights in the range $[0,1]$ to each input variable. Input variables with higher discriminating power have their values set close to 1, while the less relevant variables have values close to 0. The weights are continuously updated during the learning process.

An evolving fuzzy linear regression tree with input selection was introduced in Ref. 7. The tree topology is incrementally adjusted using a statistical test that enables updating the number of tree nodes and of input variables as new data are input.

This paper extends the X-eNFN-AFS (*eXtended Evolving Neural Fuzzy Network with Adaptive Feature Selection*) approach originally introduced in Ref. 9. The X-eNFN-AFS expands the evolving neural network constructed with neo-fuzzy neurons (NFN)[10] suggested in Refs. 3, 11 and 12.

A neural fuzzy network assembled with neo-fuzzy neurons and a scheme for adaptive input selection was first introduced in Ref. 3. Called NFN-AFS (*Neural Fuzzy Network with Adaptive Feature Selection*), the input selection scheme starts with one or more input variables and, using the input data stream and a statistic test, decides if a new input variable should be added, and if an existing variable should be maintained or excluded. The number of fuzzy sets that granulate the input variables domains is chosen *a priori* and kept the same during operation.

Later, Ref. 11 developed an evolving fuzzy network with neo-fuzzy neurons called eNFN (*Evolving Neural Fuzzy Network*). The eNFN uses an incremental learning procedure to add or delete membership functions simultaneously with weights update. The learning procedure uses input data to estimate current modeling error and verify if adaptation should proceed varying the number of membership functions for each input.

Next, an evolving neural fuzzy network with adaptive input selection eNFN-AFS (*Evolving Neural Fuzzy Network with Adaptive Feature Selection*) was built[12]. Essentially, eNFN-AFS combines the two previous approaches, namely NFN-AFS and eNFN. The eNFN-AFS adaptation proceeds by including or excluding input variables, and either add or exclude membership functions. Adaptation is done simultaneously with adjustments of the neural network weights. eNFN-AFS granulates the input variables domains choosing membership functions from a fixed set of membership functions, one set of each input variable.

Differently from eNFN-AFS, the X-eNFN-AFS approach addressed in this paper uses input data to granulate the input variables domains of a *current* and a *candidate* model. Both, *current* and *candidate* models have an associated set of membership functions for each of the corresponding input variables and, similarly as in the previous approaches, input variables selection is done using the modeling error and a statistical test.

Variable selection, input domains adaptation, and weights are updated simultaneously.

The remaining of the paper is organized as follows. After this introduction, Section 2 details the evolving learning algorithm with adaptive input selection suggested herein. Section 3 addresses prediction and nonlinear system identification application examples, and evaluates and compares the performance of X-eNFN-AFS against state of the art evolving modeling approaches. Section 4 concludes the paper with a summary of its contributions and suggestions for further studies.

2. Evolving Neural Fuzzy Network with Adaptive Input Selection

Figure 1 depicts the structure of the X-eNFN-AFS neural network. The input variables at t are $x_{t1},..,x_{tn}$, the individual outputs are denoted by $y_{t1},..,y_{tn}$, the network weights are $q_{i1},..,q_{imi}$, and the network output is $\hat{y}_t$.

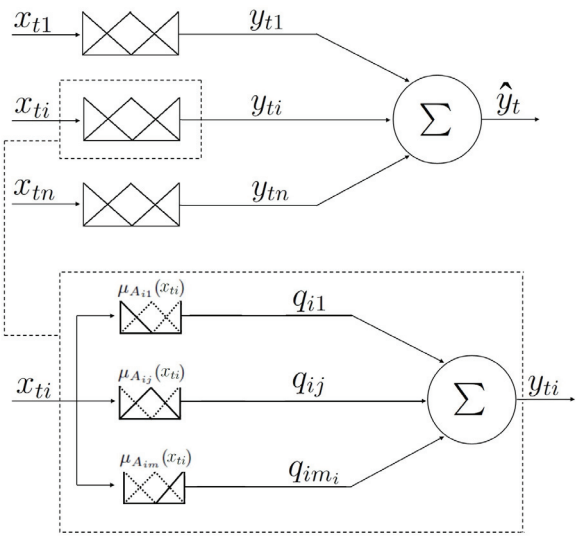

Figure 1. Structure of the X-eNFN-AFS neural.

The X-eNFN-AFS uses an incremental learning algorithm to select the inputs, to evolve the network structure, and to adjust the neural network weights concurrently to produce an output. Computations in each of these steps are recursive and there is no need to store past data. The input selection step uses a statistical test to decide if a new variable should be added, or if an existing variable should be removed or maintained. The network structure evolves by adding or deleting a membership function based using the input data and the modeling error. The weights of the neural network are updated using one-step gradient descent algorithm with optimal learning rate. An overview of the steps of the learning algorithm for X-eNFN-AFS is as follows:

- Choose initial input variables and set *current* and *candidate* models. This step is performed only once to start the algorithm.
- Choose initial parameters of the membership functions, i.e. their modal values b_i. This step is performed only once from the lower ($\min x_i$) and upper ($\max x_i$) bounds of the input variables domains.
- Check if the input x_{ti} is greater than the upper bound ($\max x_i$) or smaller than lower bound ($\min x_i$). Decide if the value of the bounds should be updated.
- Compute the membership degrees μA_{ij} of input x_{ti}, find the most active membership function and update its modal value.
- Compute the neural network output $\hat{y}_t$.
- Update the neural network weights q_{ij} .
- Choose the best *candidate* model. Decide if the best *candidate* model should replace the *current* model.
- Check whether the most active membership function represents well the neighborhood of input x_{ti} . Decide if a new membership function should be created to refine the neighborhood of x_{ti} .
- Find the oldest inactive membership function. Decide if this membership function should be removed.

The details of each of these steps are given next.

2.1. *Step 1: Initial Input Variables and Initial Current and the Candidate Models*

Variable selection requires pre-selection of a set of input variables. One or more variables of this set are selected to start learning. The initial set of input variables can be constructed either from *a priori* knowledge, or employing ranking methods[12].

Input variable selection considers the *current* model and two *candidate* models. The first *candidate* model is constructed adding new variables to the *current* model. The second *candidate* model is constructed excluding an existing variable from the *current* model. The idea is to check if it is worth replacing the *current* model by either a more complex (first) or simpler (second)

candidate models, the one which improves modeling performance.

Let n be the total number of input variables and a be the number of input variables of the *current* model. Thus, we can choose any of the remaining $(n-a)$ variables and add them in the *current* model to assemble a *candidate* model with $(a+1)$ input variables. On the contrary, any of the a variables of the *current* model can be removed to obtain a simpler *candidate* model with $(a-1)$ input variables. Figure 2 illustrates the idea. The set of input variables $\{x_1, x_2, x_3\}$ has $n = 3$ elements. The *current* model (highlighted in red) has two input variables x_1, and x_2, and $a = 2$. The first *candidate* model has the variable x_3 as an added input. There are two *candidate* models: the first has x_2 as input (x_1 removed) and the second has x_1 as input (x_2 removed).

2.2. *Step 2: Initialization Membership Functions*

Initially the domain of each input variable is uniformly partitioned using triangular membership functions. Triangular membership functions are defined by their modal values, and by the lower and upper bounds of their support. We denote the modal value of the k-th membership function (and refer to the membership function itself) by b_k. The lower bound of its support is at the modal value of the $(k-1)$-th adjacent membership function b_{k-1}, and upper bound of its support is at the modal of the $(k+1)$-th adjacent membership function b_{k+1}. The initial number of membership functions can be chosen empirically, based on *a priori* knowledge, or using a clustering technique[14]. In this paper, initially *current* and *candidate* models start with two membership functions for each input variable. The modal values of the initial membership functions are chosen as follows:

$$b_{i1=\min x_i}, \quad b_{i2=\max x_i}, \tag{1}$$

where i indexes the input variable, $\min_{x_i}$ is the lower bound, and $\max_{x_i}$ the upper bound of the i-th input variable domain. Adding and/or removing membership functions, depending on the input data and modeling error, refine granulation of the input domain. The procedures to add and to delete a membership function are detailed in Sections 2.8 and 2.9, respectively.

Steps 1 and 2 are performed only once and starts the X-eNFN-AFS learning and adaptation.

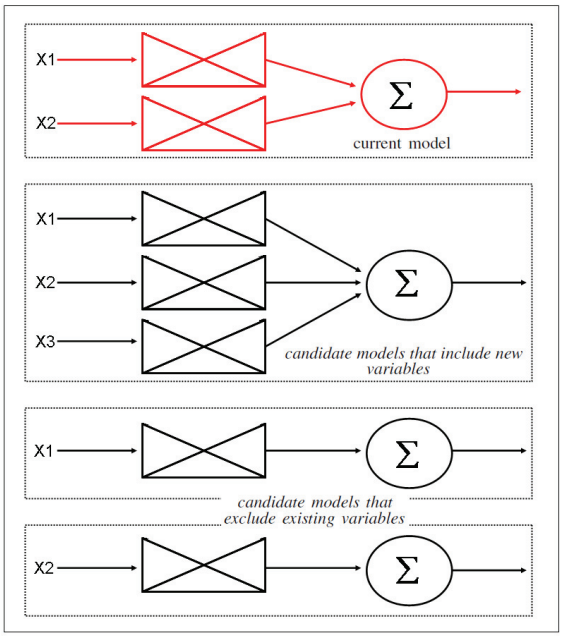

Figure 2. *Candidate* models.

2.3. *Step 3 - Context Adaptation*

In data stream-based applications there may be changes in operating conditions that enable the emergence of data whose values are outside the $\min_{x_i}$ and $\max_{x_i}$ bounds. Therefore, it is important to update the bounds of the input variables. A simple way to update $\min_{x_i}$ and $\max_{x_i}$ is as follows:

If $x_{ti} < \min_{x_i}$, then

$$\min_{x_i} = x_{ti} \text{ and } b_{i1} = \min_{x_i}, \tag{2}$$

If $x_{ti} > \max_{x_i}$, then

$$\max_{x_i} = x_{ti} \text{ and } b_{im_i} = \max_{x_i}, \tag{3}$$

in other words, the maximum and minimum bounds for each input variable are updated as the algorithm receives new samples that exceed current bounds. This step is performed whenever new data are input.

2.4. *Step 4 - Modal Value Update*

This step updates the modal value of the most active membership function enabled by current input x_{ti}, $i = 1,\ldots, n$. It works as follows. For each input variable, let b_i^* be the index of the most active membership

function. The membership function indexed by b_i^* is updated according to the following rules:

If $b_i^* > 1$ and $b_i^* \neq m_i$, then

$$b_{ib_i^*}^{new} = b_{ib_i^*}^{old} + \beta\left(x_{ti} - b_{ib_i^*}^{old}\right), \tag{4}$$

where β is a learning rate chosen empirically. A typical value, the one adopted in this paper, is $\beta = 0.01$.

If $b_i^* = 1$ or $b_i^* = m_i$, then

the modal value is kept the same because in this case the modal values corresponds to upper and lower bounds, respectively.

2.5. *Step 5: Model Output*

Each model (*candidate* or *current*) mirrors a set of zero-order Takagi Sugeno (TS)[15] rule-based model, one for each input variable. We detail only the procedure for the *current* model for short. The procedure for the *candidate* models is, *mutatis mutandis*, the same. The output $\hat{y}_t$ of the model at step t is the sum of individual outputs of the *current* model, i.e.:

$$\hat{y}_t = \sum_{i=1}^{a} y_{ti}, \tag{5}$$

where a is the number of input variables for the *current* model.

The domain of each input variable x_{ti} is granulated using m_i membership functions. Because partitions are uniform, at most two of the membership functions are active for a given input x_{ti} (Figure 3).

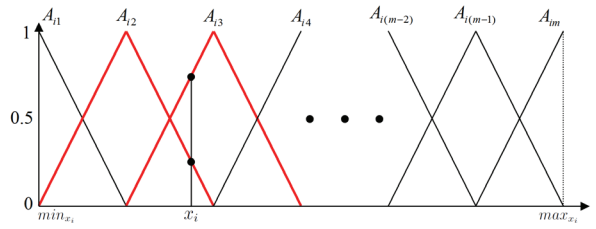

Figure 3. Uniform partition and active membership functions.

The individual outputs y_{ti} are computed using the active membership functions only, that is:

$$y_{ti} = \mu A_{ik_i}(x_{ti}) q_{ik_i} + \mu A_{ik_i+1}(x_{ti}) q_{ik_i+1}, \tag{6}$$

where $\mu A_{ik_i}(x_{ti})$ and $\mu A_{ik_i+1}(x_{ti})$ are the activation degree of membership functions A_{ik_i} and A_{ik_i+1}, k_i and k_i+1 indexes the active membership functions, and q_{ik_i} and q_{ik_i+1} are the network connection weights.

2.6. *Step 6: Network Connection Weights Update*

Only membership functions A_{ik_i} and A_{ik_i+1} are active for each input x_{ti} and only the corresponding connection weights are updated. The updating mechanism uses a gradient-descent mechanism:

$$\begin{aligned} q_{ik_i} &= q_{ik_i} - \alpha_t (y_t - \hat{y}_t) \mu A_{ik_i}(x_{ti}), \\ q_{ik_i+1} &= q_{ik_i+1} - \alpha_t (y_t - \hat{y}_t) \mu A_{ik_i}(x_{ti}), \end{aligned} \tag{7}$$

where y_t is the desired output, $\hat{y}_t$ is the network output, and α_t is the learning rate. In this paper we adopt a closed formula to compute the value of α_t that gives zero error at each learning step. The optimal learning rate[14] is:

$$\alpha_t = \frac{1}{\sum_{i=1}^{n} \mu A_{ik_i}(x_{ti})^2 + \mu A_{ik_i+1}(x_{ti})^2}. \tag{8}$$

Steps 4, 5, and 6 are repeated for all *candidate* models.

2.7. *Step 7: Adaptive Input Variable Selection*

Adaptive input variable selection is based on the F test[16]. The F test evaluates the quality of models, considering their accuracy and number of free parameters. Two models are evaluated, one simpler and other more complex than the *current* model. Here, model complexity refers to the number of input variables and the number of membership functions of the model. F test analyzes the cost-benefit between more precise and more complex models.

The F test[16] uses the following statistic:

$$F = \frac{(RSS_a - RSS_c)(S - p_c)}{RSS_c(p_c - p_a)}, \tag{9}$$

where S is the number of samples used to estimate the parameters of the models, RSS_a and RSS_c are, respectively, the sum of squared residuals for the *current* and *candidate* model, and p_c and p_a are the number of free parameters of each model. The number of parameters p is f the number of input variables times

the number of membership functions. Assuming that residuals distribution is normal, F follows a Fisher distribution with ($p_c - p_a, S - p_c$) degrees of freedom.

In the F test the model parameters are estimated using the same samples, but the number of samples used to estimate the parameters may not be the same at each learning step. This is because new *candidate* models are created whenever the *current* model is replaced by a *candidate* model. The new *current* model continues with the same number of parameters and statistics, but the new *candidate* models start from scratch. Therefore, the number of samples used to estimate the *current* model parameters will always be equal to or greater than the number of samples used for the *candidate* models.

A modification of the e F test suggested in Ref. 17 is particularly attractive for incremental algorithms. In Refs. 7, 17 and 18 the F test is used in incremental linear regression tree learning. This variation of the F test is used in this work to compare the *current* model with the *candidate* models.

The statistic F_{inc} (10) of the *candidate* model that has a new variable added to the *current* model is computed as follows:

$$F_{inc} = \frac{(RSS_a - RSS_c)(S_c - p_c)}{RSS_c(S_c - S_{a+}p_a - p_c)}, \tag{10}$$

where S_a e S_c are the number of samples used to estimated the parameters of the *current* and *candidate* model, respectively. F_{inc} follows a Fisher distribution with ($S_c - S_{a+}p_a - p_c, S_c - p_c$) degrees of freedom.

The statistic F_{exc} (11) is used by the *candidate* model that has a variable removed from the *current* model. F_{exc} follows a Fisher distribution with ($S_c - S_{a+}p_c - p_a, S_c - p_c$) degrees of freedom.

$$F_{exc} = \frac{(RSS_a - RSS_c)(S_c - p_c)}{RSS_c(S_c - S_{a+}p_c - p_a)}. \tag{11}$$

The F_{inc} and F_{exc} statistics requires *p_values* to be found for all *candidate* models. The c*andidate* with the smallest *p_value* is the best *candidate* model, and it replaces the *current* model only if its *p_value* is smaller than a significance level γ . Because the hypothesis test is done λ times using the same data, it is necessary to consider multiple comparison approaches[17] such as the Bonferroni scheme. Therefore, the significance level must be divided by the number of tests, and a *candidate* model replaces the *current* model only

$$\text{If } p\_value < \frac{\gamma}{\lambda} \tag{12}$$

where λ is $(n-a)$ for F_{inc} and a for F_{exc} . Typical values for the significance level γ are 0.01 and 0.05 .

The adaptive input variable selection algorithm can be summarized as follows.

Procedure Input_Selection
nc: number of *candidate* models
For l = 1 : *nc*
  compute F^l
  compute p_value^l
end For
find *candidate* model with smallest *p_value*
CreateExclude=1
Model replacement test
If $p\_value^z < \gamma / \lambda$
  CreateExclude=0
  replace the *current* model by the *candidate* model *z*
  current model keeps statistics/parameters of *z*
  set new *candidate* models
end If

2.8. *Step 8: Creation of the Membership Functions*

Creation of membership functions aims to refine input domains granulation, and to reduce the output error uniformly. Granulation of input domains is performed using the error caused by the currently active membership functions. The mean value of the local output errors of the rules corresponding to the active membership functions are compared against the mean value of the global modeling error. If the local mean error value is greater than the overall mean error, then the local region is refined adding new membership function as follows.

For each new input x_t , the mean value $\hat{\mu}_{g_t}$ and variance $\hat{\sigma}^2_{g_t}$ of the overall modeling error are found as follows:

$$\hat{\mu}_{g_t} = \hat{\mu}_{g_{t-1}} - \beta(\hat{\mu}_{g_{t-1}} - (\hat{y}_t - y_t)), \tag{13}$$

$$\hat{\sigma}^2_{g_t} = (1-\beta)(\hat{\sigma}^2_{g_{t-1}} + \beta(\hat{\mu}_{g_t} - (\hat{y}_t - y_t))^2). \tag{14}$$

Similarly, for the input x_t the local mean error $\hat{\mu}_{b_{ti}^*}$ corresponding to the most active membership function b_i^* is computed recursively using:

$$\hat{\mu}_{b_{ti}^*} = \hat{\mu}_{b_{t-1i}^*} - \beta(\hat{\mu}_{b_{t-1i}^*} - (\hat{y}_t - y_t)). \quad (15)$$

To prevent excessively fine granulation, a threshold τ is used to limit the smallest distance between the modal values of adjacent membership functions. If $b_i^* > 1$ and $b_i^* \neq m_i$, then the distance is found using:

$$dist = \frac{\left(b_i b_i^* - b_i b_i^* - 1\right)}{3}. \quad (16)$$

On the contrary, if the most active function is such that $b_i^* = 1$ or $b_i^* = m_i$, then the *dist* is computed by (17) and (18), respectively.

$$dist = \frac{\left(b_i b_i^* - b_i b_i^*\right)}{2}. \quad (17)$$

$$dist = \frac{\left(b_i b_i^* - b_i b_i^* - 1\right)}{2}. \quad (18)$$

The number of rules is not fixed *a priori* and it depends of the learning process and data only. This mechanism avoids complex models and overfitting. Indirectly, the threshold τ controls of the number of rules. Limit τ is computed using:

$$\tau = \frac{(\max x_i - \min x_i)}{\eta}, \quad (19)$$

where η is a user-defined parameter. Typically $\eta \in [5, 25]$.

A membership function is created and added:

$$\text{If } \hat{\mu}_{b_{ti}^*} > \hat{\mu}_{g_t} + \hat{\sigma}^2_{g_t} \text{ and } dist > \tau. \quad (20)$$

Insertion of a new membership function requires updating the granulation of the *i*-th input variable domain. This can be done as follows:

If $b_i^* > 1$ and $b_i^* \neq m_i$, then
the most active function is replaced by two new membership functions whose modal values are found from (21) and (22).

$$b_{1new1} = b_{ib_i^*-1} + dist. \quad (21)$$

$$b_{1new1} = b_{ib_i^*-1} + 2 * dist. \quad (22)$$

If $b_i^* = 1$, then
a new membership function is inserted between the first and second, and its modal value is found by:

$$b_{new} = b_{ib_i^*} + dist. \quad (23)$$

If $b_i^* = m_i$, then
the new membership function is inserted between the last and the previous, with modal value computed by:

$$b_{new} = b_{ib_i^*} - dist. \quad (24)$$

The procedure Create_Function summarizes the mechanism to create and add membership functions.

Procedure Create_Function
find b_i^*
compute $\hat{\mu}_{g_t}$, $\hat{\sigma}^2_{g_t}$, $\hat{\mu}_{b_{ti}^*}$, τ, $dist$
If $\hat{\mu}_{b_{ti}^*} > \hat{\mu}_{g_t} + \hat{\sigma}^2_{g_t}$ and $dist > \tau$
create and add new function
update parameters
end If

2.9. *Step 9: Exclusion of the Membership Functions*

This step is a mechanism to reduce the number of membership functions using the concept of age[19, 20]. The idea of age is used to determine for how many steps a membership function has been inactive. The age of a membership function is:

$$age_i = t - a_i, \quad (25)$$

where a_i is the step at which *i*-th membership function turn active first and t is the current step.

The scheme to exclude a membership function is as follows. For each input variable i, find b_i^-, the index of the least active membership function enabled by x_{ti}. The membership function indexed by b_i^- is excluded

$$\text{If } age_{b_i^-} > \omega \text{ and } m_i > 2, \quad (26)$$

where ω is a threshold. Typically, the value of ω is chosen between 50 and 250.

After a membership function is excluded, input domains granulation is updated as follows:

If $b_i^- > 1$ and $b_i^- \neq m_i$, then

the function is excluded and the lower and upper bounds of the adjacent membership functions adjusted. Both, upper and lower bounds change to keep partition uniform.

If $b_i^- = 1$, then

the function is excluded and the modal values of the adjacent membership functions are set as $\min x_i$.

If $b_i^- = m_i$, then

the function is excluded and the modal values of the adjacent membership functions are set as $\max x_i$.

The procedure to exclude membership functions is:

Procedure Exclude_Function
update age_i
find b_i^-
If ($age_{b_i^-} > \omega$) and ($m_i > 2$),
remove membership function indexed b_i^-
update parameters
end If

Steps 8 and 9 are repeated for all *candidate* models.

2.10. *X-eNFN-AFS Learning Algorithm*

The steps the X-eNFN-AFS learning algorithm can be summarized as follows.

Inputs x_t, y_t, γ, β, η, ω
Output $\hat{y}_t$
initialize b_{ij}
set initial *current* and *candidate* models
For t = 1, 2,...
input x_t, y_t
check context adaptation
***Current* Model**
compute $\mu A_{ik_i}(x_{ti})$, $\mu A_{ik_i+1}(x_{ti})$
compute $\hat{y}_t$
compute α_t
update b_{ij}, q_{ik_i}, q_{ik_i+1}, RSS_a, S_a, p_a
***Candidate* Models**
nc: number of *candidate* models
For l = 1:*nc*
compute $\mu A_{ik_i}(x_{ti})$, $\mu A_{ik_i+1}(x_{ti})$
compute $\hat{y}_t^l$
compute α_t^l
update b_{ji}^l, $q_{ik_i}^l$, $q_{ik_i+1}^l$, RSS_c^l, S_c^l, p_c^l
end For
//Procedure Input_Selection
If CreateExclude=1
nm : number of models (*current* and c*andidates)*
For l=1:*nm*
For i = 1: n
Procedure Create_Function
Procedure Exclude_Function
end For
end For
end If
end For

3. Computational Results

In this section, the evolving neural fuzzy network with adaptive input selection X-eNFN-AFS is evaluated and compared with other six approaches representative of the state of the art in evolving fuzzy systems modeling, namely: DENFIS[21], eMG[22], eNFN[11], eNFN-AFS[12], eTS[23] and xTS[24]. All approaches are evaluated using prediction and nonlinear system identification examples. Simulations process data as a stream. The parameters and the structure of the models evolve as each data sample is input.

The dataset is split into two subsets with 50% of the samples each. The first subset is used to find the best parameters of the models using exhaustive search whereas the second is used to evaluate the performance of the models. The best parameters, i.e., the parameters values that produce the lowest modeling error, are used for performance evaluation. The modeling error measure adopted is the Root Mean Squared Error (*RMSE*):

$$RMSE = \left(\frac{1}{N} \sum_{t=1}^{N} (y_t - \hat{y}_t)^2 \right)^{\frac{1}{2}}, \quad (27)$$

where N is the number of samples, y_t is the desired output, and $\hat{y}_t$ is the model output.

3.1. *Predicting the Position of a Magnetic Levitation Sphere*

Performance evaluation is done using models to predict the position of a sphere of a magnetic levitation system (MagLev). The position of the sphere is one of the state variables whose value depends on the voltage applied to the coil that produces the magnetic field. The data set used in simulations was extracted from an actual magnetic levitation system[25, 26]. The MagLev was run for 60 seconds with a sampling rate of 10^{-3} seconds (1 ms), resulting in a total of 60000 samples. Initially the desired position is a sine function whose magnitude is 0.5 and frequency 0.5 Hz. At t = 17 seconds the desired position becomes a square wave with 0.4 of the magnitude and 0.5 Hz, a step function with magnitude between -1 and –2 at t = 31 seconds, the sine function again at t = 41, and finally the square function after t = 51. The purpose to change the desired position as described above is to evaluate the behavior of the evolving model when operating condition changes.

The aim of the computational experiments is to use the model to predict the position of the sphere one step ahead. The model has the following form[25]:

$$\hat{y}_t = f(dp_{t-5}, mp_{t-5}, mp_{t-4}, mp_{t-3}), \quad (28)$$

where $\hat{y}_t$ is the model output at t, dp is the desired position, mp is the measured position.

A total of 60000 samples were produced, 3000 to estimate the parameters, and 30000 to evaluate the performance of all models. Models with adaptive input selection eNFN-AFS and X-eNFN-AFS start with all four inputs. DENFIS, eMG, eNFN, eTS, and xTS start with and keep all four inputs dp_{t-5}, mp_{t-5}, mp_{t-4}, mp_{t-3}, respectively. The actual position and the X-eNFN-AFS prediction are shown in Figure 4.

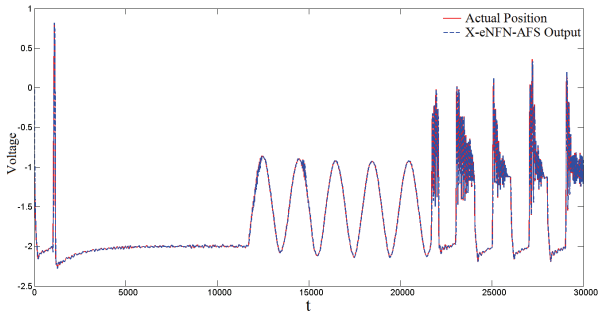

Figure 4. X-eNFN-AFS prediction of the sphere position.

Figure 5 illustrates how the X-eNFN-AFS evolves its structure and selects the input variables as each new sample is input. The adaptive input variable selection scheme of the X-eNFN-AFS selected ($dp_{t-5}, mp_{t-5}, mp_{t-4}$) as inputs.

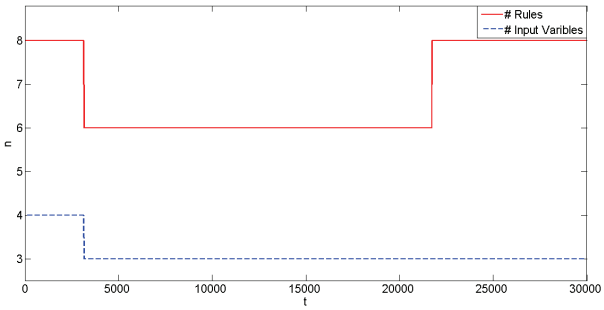

Figure 5. Structure evolution and variable selection of X-eNFN-AFS for the MagLev system.

The RMSE performance, the number of input variables, and the number of rules of the modeling approaches evaluated are summarized in Table 1. The best performance is achieved by X-eNFN-AFS followed by eNFN-AFS, eNFN and eMG. The performance of the X-eNFN-AFS, eNFN-AFS, eNFN and eMG are higher than DENFIS, eTS and xTS by one order of magnitude. The best values of the models parameters found through exhaustive search are shown in Table 2.

Table 1. Sphere position prediction performance.

Model	Input Variables	Number of rules	RMSE
DENFIS	04	13	1.5815
eMG	04	11	0.0832
eNFN	04	09	0.0737
eNFN-AFS	03	06	0.0731
eTS	04	02	1.7912
X-eNFN-AFS	03	08	0.0711
xTS	04	02	1.7892

Table 2. Best model parameters: sphere position prediction.

Model	Parameters
DENFIS	$dthr$=0.1, $mofn$=4
eMG	λ=0.05, ω=20, α = 0.01, $\Sigma_{init}=10^{-1}I_{11}$
eNFN	β=0.01, ω=100, η=10,
eNFN-AFS	β=0.01, ω=100, η=10, γ = 0.05
eTS	r=0.04, Ω=750
X-eNFN-AFS	β=0.01, ω=100, η=10, γ = 0.05
xTS	Ω=750

3.2. *System Identification*

In this section the evolving modeling approach is evaluated using a system identification problem. The nonlinear system[11, 22] to be modeled is:

$$y_t=\frac{\sum_{i=1}^{m} y_{t-i}}{1+\sum_{i=1}^{m}(y_{t-i})^2}+u_{t-i}, \quad (29)$$

where $u_t=\sin(2\pi t/20)$, $y_j=0$ for $j=1,...,m$ to $m=10$. The aim is to use the model to predict the current output y_t using past inputs and outputs. The model has the following form[22]:

$$\hat{y}_t=f(y_{t-1},y_{t-2},...,y_{t-9},y_{t-10},u_{t-1}), \quad (30)$$

where $\hat{y}_t$ is the model output at t.

Here a total of 3300 samples, 1650 to estimate, and the remaining 1650 to evaluate the performance of all model. Figure 6 illustrates the actual and X-eNFN-AFS model outputs.

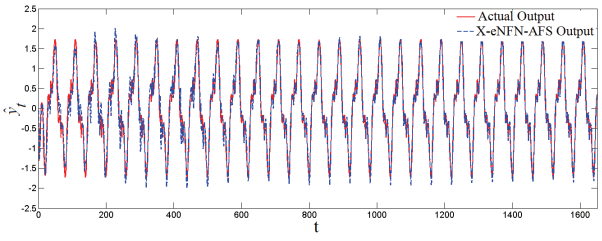

Figure 6. Nonlinear system identification.

Figure 7 shows how the structure of X-eNFN-AFS modifies as data are input. The adaptive input selection scheme of X-eNFN-AFS selected (y_{t-1}, y_{t-2}, y_{t-3},y_{t-5},y_{t-6},y_{t-7},y_{t-8},y_{t-10}) as input variables.

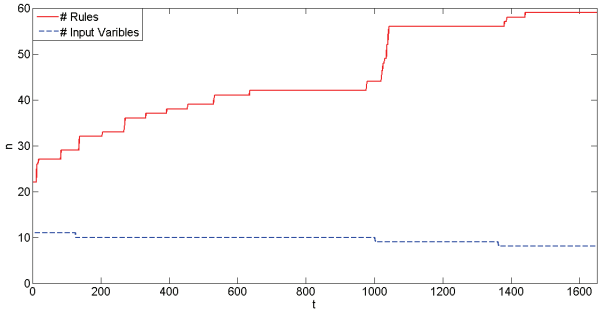

Figure 7. Structure evolution and variable selection of X-eNFN-AFS for nonlinear system identification.

Table 3 shows the RMSE, the number of input variables, and the number rules after simulation ends. The X-eNFN-AFS has similar performance as eMG, eNFN and eNFN-AFS, and better performance than DENFIS, eTS and xTS. The best parameters of the models are given in Table 4.

Table 3. Modeling performance for nonlinear system identification.

Model	Input Variables	Number of rules	RMSE
DENFIS	11	13	0.2080
eMG	11	10	0.1244
eNFN	11	107	0.1210
eNFN-AFS	08	65	0.1401
eTS	11	09	0.8303
X-eNFN-AFS	08	59	0.1234
xTS	11	03	0.8316

Table 4. Best model parameters: nonlinear system identification.

Model	Parameters
DENFIS	$dthr$=0.1, $mofn$=4
eMG	λ=0.05, ω=10, α=0.01, $\Sigma_{init}=10^{-1}I_{11}$
eNFN	β=0.01, ω=100, η=15
eNFN-AFS	β=0.01, ω=100, η=10, γ = 0.05
eTS	r=0.04, Ω=750
X-eNFN-AFS	β=0.01, ω=100, η=10, γ = 0.05
xTS	Ω=750

4. Conclusion

This paper has suggested an approach for adaptive modeling with input variable selection using neo-fuzzy neural network called X-eNFN-AFS. The X-eNFN-AFS uses a learning procedure that simultaneously selects

the input variables, adapts the granulation of the input variables domains, and updates the parameters of the neural network. The approach uses *current* and *candidate* models of distinct complexity, input data, and the statistic *F* test to select model inputs.

Prediction and nonlinear system identification application problems were used to evaluate and compare the X-eNFN-AFS against current state of the art evolving modeling approaches. Simulation results indicate that the X-eNFN-AFS has comparable or better performance than the remaining evolving models.

Future work shall consider the dependencies between input variables, extend the network for multiple outputs, and investigate mechanisms to reduce the complexity of the input selection algorithm. Mechanisms to automatically select user-defined parameters to turn X-eNFN-AFS more autonomous are also important for future work.

Acknowledgements

The authors acknowledge the support Brazilian Ministry of Education (CAPES), the Brazilian National Research Council (CNPq), and the Research Foundation of the State of Minas Gerais (FAPEMIG) for their support. The last author is grateful to CNPq for grant 305906/2014-3.

References

1. E. Lughofer, On-line assurance of interpretability criteria in evolving fuzzy systems - achievements, new concepts and open issues, Information Sciences 251 (0) (2013) 22-46. doi:10.1016/j.ins.2013.07.002.
2. S. Tung, C. Quek, C. Guan, eT2FIS: An evolving Type-2 neural fuzzy inference system, Information Sciences 220 (0) (2013) 124-148. doi:10.1016/j.ins.2012.02.031.
3. A. Silva, W. Caminhas, A. Lemos, F. Gomide, Evolving neural fuzzy network with adaptive feature selection, in: Proceedings of the 11th International Conference on Machine Learning and Applications, ICMLA '12, Vol. 2, 2012, pp. 440-445. doi:10.1109/ICMLA.2012.184.
4. J. Zhu, N. Lao, E. Xing, Grafting-light: fast, incremental feature selection and structure learning of Markov random fields, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '10, 2010, pp. 303-312. doi:10.1145/1835804.1835845.
5. I. Katakis, G. Tsoumakas, I. Vlahavas, Dynamic feature space and incremental feature selection for the classification of textual data streams, in: Proceedings of the International Workshop on Knowledge Discovery from Data Streams, ECML/PKDD '06, 2006, pp. 107-116.
6. E. Lughofer, On-line incremental feature weighting in evolving fuzzy classifiers, Fuzzy Sets Systems 163 (1) (2011) 1-23. doi:10.1016/j.fss.2010.08.012.
7. A. Lemos, W. Caminhas, F. Gomide, Evolving fuzzy linear regression trees with feature selection, in: Proceedings of the IEEE Workshop on Evolving and Adaptive Intelligent Systems, EAIS '11, Vol. 1, 2011, pp. 31-38. doi:10.1109/EAIS.2011.5945919.
8. E. Lughofer, P. Angelov, X. Zhou, Evolving single- and multi-model fuzzy classifiers with FLEXFIS-Class, in: Proceedings of the IEEE International Fuzzy Systems Conference, FUZZ-IEEE '07, 2007, pp. 1-6. doi:10.1109/FUZZY.2007.4295393.
9. A. Silva, W. Caminhas, A. Lemos, F. Gomide, Extended Approach for Evolving Neo-Fuzzy Neural with Adaptive Feature Selection, in Decision Making and Soft Computing, Ch. 107, pp. 651-656. doi:10.1142/9789814619998 0107.
10. T. Yamakawa, E. Uchino, T. Miki, H. Kusabagi, A neo fuzzy neuron and its applications to system identification and predictions to system behavior, in: Proceedings of the International Conference on Fuzzy Logic and Neural Networks, Vol. 1, 1992, pp. 477-484.
11. A. Silva, W. Caminhas, A. Lemos, F. Gomide, A fast learning algorithm for evolving neo-fuzzy neuron, Applied Soft Computing 14, Part B (0) (2014) 194-209. doi:10.1016/j.asoc.2013.03.022.
12. A. Silva, W. Caminhas, A. Lemos, F. Gomide, Evolving neo-fuzzy neural network with adaptive feature selection, in: Proceedings of the 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI CBIC), 2013, pp. 341-349. doi:10.1109/BRICS-CCI-CBIC.2013.64.
13. I. Guyon, A. Elisseeff, An introduction to variable and feature selection, The Journal of Machine Learning Research 3 (2003) 1157-1182.
14. W. Caminhas, F. Gomide, A fast learning algorithm for neofuzzy networks, in: Proceedings of the Information Processing and Management of Uncertainty in Knowledge Based Systems, IPMU '00, Vol. 1, 2000, pp. 1784-1790.
15. T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Transactions on Systems, Man and Cybernetics 15 (1) (1985) 116-132.
16. M. Allen, Understanding Regression Analysis, 1st Edition, Springer, 1997.
17. D. Potts, C. Sammut, Incremental learning of linear model trees, Machine Learning 61 (1) (2004) 5-48. doi:10.1007/s10994-005-1121-8.
18. A. Lemos, W. Caminhas, F. Gomide, Fuzzy evolving linear regression trees, Evolving Systems 2 (2011) 1-14. doi:10.1007/s12530-011-9028-z.

19. P. Angelov, D. Filev, Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models, in: Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE '05, 2005, pp. 1068-1073. doi:10.1109/FUZZY.2005.1452543.
20. E. Lughofer, P. Angelov, Handling drifts and shifts in on-line data streams with evolving fuzzy systems, Applied Soft Computing 11 (2) (2011) 2057-2068. doi:10.1016/j.asoc.2010.07.003.
21. N. Kasabov, Q. Song, DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Transactions on Fuzzy Systems 10 (2) (2002) 144-154. doi:10.1109/91.995117.
22. A. Lemos, W. Caminhas, F. Gomide, Multivariable gaussian evolving fuzzy modeling system, IEEE Transactions on Fuzzy Systems 19 (1) (2011) 91-104. doi:10.1109/TFUZZ.2010.2087381.
23. P. Angelov, D. Filev, An approach to online identification of Takagi-Sugeno fuzzy models, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 34 (1) (2004) 484-498. doi:10.1109/TSMCB.2003.817053.
24. P. Angelov, X. Zhou, Evolving fuzzy systems from data streams in real-time, in: Proceedings of the International Symposium on Evolving Fuzzy Systems, 2006, pp. 29-35. doi:10.1109/ISEFS.2006.251157.
25. A. Silva, W. Caminhas, A. Lemos, F. Gomide, Rede neuro-fuzzy evolutiva para estimação em tempo real de posição de um sistema de levitação magnética (in portuguese), in: XI Simpósio Brasileiro de Automação Inteligente, XI SBAI, 2013, pp. 1-8.
26. I. Feedback, Magnetic levitation control experiments, 33-942s, UK.